**Approach, design & algorithm** – I’m writing this as I’m reading through the Project materials, and while coding for the project

* Read the Project-1C Write-up.docx
* Read the PasswordCheckerUtility.html doc
  + I also found many superfluous files in the same folder, containing the same pieces of information from the PasswordCheckerUtility.html
* Created custom Exceptions classes from the Write-up using the provided LengthException class as a template.
  + InvalidSequenceException.java
  + NoDigitException.java
  + NoLowerAlphaException.java
  + NoSpecialCharacterException.java
  + NoUpperAlphaException.java
  + UnmatchedException.java
    - The custom exception message from the doc said: “The passwords do not match”.   
      However, I changed it to “Passwords do not match” from the *PasswordCheckerTestPublic.java*
  + WeakPasswordException.java
* Created ***PasswordCheckerUtility*** class
* Added the following *public static boolean* methods to ***PasswordCheckerUtility*** from the PasswordCheckerUtility.html doc.
  + *comparePasswordsWithReturn(String password, String passwordConfirm)*
  + *comparePasswords(String password, String passwordConfirm)*
  + *isValidLength(String password)*
  + *hasUpperAlpha(String password)*
  + *hasLowerAlpha(String password)*
  + *hasDigit(String password)*
  + *hasSpecialChar(String password)*
  + *NoSameCharInSequence(String password)*
  + *isValidPassword(String password)*
  + *getInvalidPasswords(ArrayList<String> passwords)*
  + *hasBetweenSixAndNineChars(String password)*
  + *isWeakPassword(String password)*
* *comparePasswordsWithReturn –* Used .*compareTo()* method to confirm that the password & passwordConfirm parameters are an exact match.
* *comparePasswords –* Used *comparePasswordsWithReturn* method to throw an UnmatchedException if needed.
* *isValidLength –* Used *.length()* method to determine if password is at least 6 characters long.
* *hasUpperAlpha, hasLowerAlpha, hasDigit –* For each of these methods, I used a for loop to cycle through each individual character in the password to find if there’s at least one uppercase letter, one lowercase letter, and one number (based on ACII values).   
  If found, the loop breaks and the method returns true.
* *hasSpecialChar –* I almost used the same code from the previous method, until I re-read the Write-up and found the provided code to check for special characters:

*Pattern pattern = Pattern.compile("[a-zA-Z0-9]\*");*

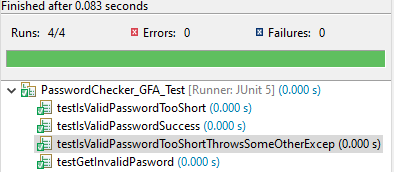
*Matcher matcher = pattern.matcher(str);*

*return (!matcher.matches());*

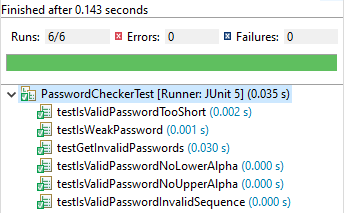
* *NoSameCharInSequence –* Used a modified for loop (stops at 2nd-to-last character in password), which checks to see if current character and next character in password are the same. If a match is found, the loop breaks and the method returns false.
* *isValidPassword* *–* After I finished coding for the above methods, I put them all under the *isValidPassword* method to verify a valid password.
* *getInvalidPasswords –* Used enhanced for loop and *isValidPassword* method with try/catch statement to catch which passwords ArrayList parameter are invalid, then adds each invalid password with its corresponding custom exception message to a new *invalidPasswords* ArrayList.
  + I struggled here because I forgot to use *.getMessage()* method to return the error message without the Exception name.
* *hasBetweenSixAndNineChars –*  similar to *isValidLength*, used *.length()* method to determine if password has least 6 characters, and 9 or fewer characters.
* *isWeakPassword –*  Uses *isValidPassword* and *hasBetweenSixAndNineChars* methods to determine if the password is valid, but weak if both methods return true.
  + This one was the trickiest for me: my code would not throw WeakPasswordException when I tried to implement *isValidPassword*.
  + But then it was somehow fixed when I was retyping the code.
* Ran the provided Junit Tests (PasswordChecker\_GFA\_Test, PasswordCheckerTest, PasswordCheckerTestPublic) to make sure all methods worked as intended.
* Then created my own PasswordCheckerTest\_STUDENT Junit test to use my own values for each method.

**Test plan & test cases**

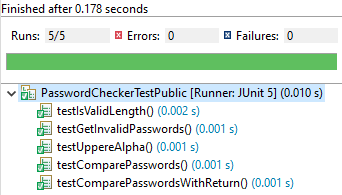
PasswordChecker\_GFA\_Test



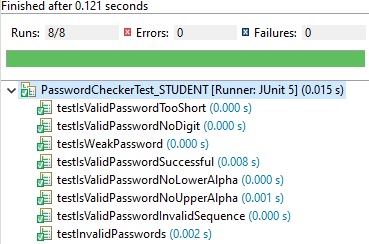
PasswordCheckerTest



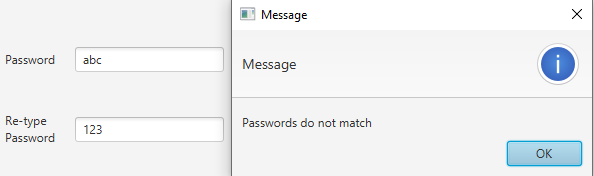
PasswordCheckerTestPublic



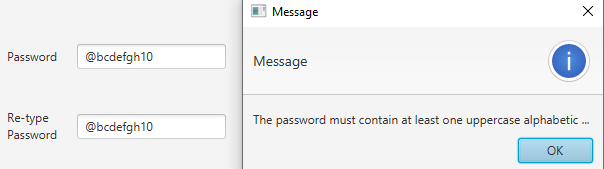
PasswordCheckerTest\_STUDENT



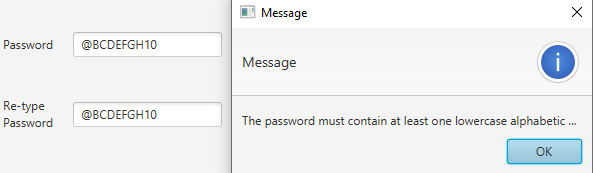
**UnmatchedException**



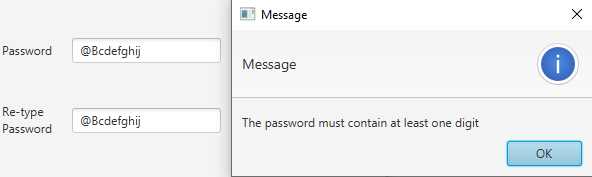
**NoUpperAlphaException.java**



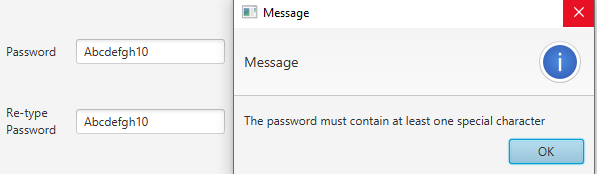
**NoLowerAlphaException**



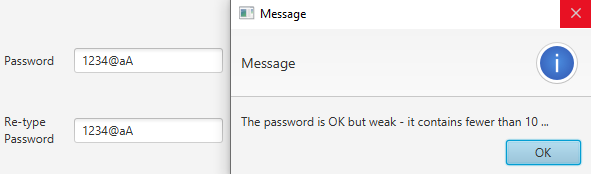
**NoDigitException**



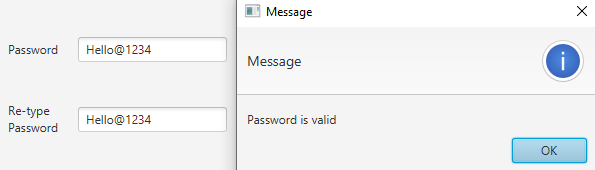
**InvalidSequenceException**



**WeakPasswordException**



**Valid Password**



**Learning Experience**

* Became more practiced in Junit tests
* Learned a lot more about using Exceptions
* Learned a little about using regex

**Assumptions**

* User will not jump straight into coding like I did.
* Provided files are written as intended
* Passwords are written only with English letters
* Having a separate file for each custom Exception (can’t have an all-encompassing custom Exception class)
* Having to throw an Exception for each missed password criteria, otherwise I might’ve used nested if statements